Mesh Shape-Quality Optimization Using the Inverse Mean-Ratio Metric*

Todd Munson†

April 20, 2004

Abstract

Meshes containing elements with bad quality can result in poorly conditioned systems of equations that must be solved when using a discretization method, such as the finite-element method, for solving a partial differential equation. Moreover, such meshes can lead to poor accuracy in the approximate solution computed. In this paper, we present a nonlinear fractional program that relocates the vertices of a given mesh to optimize the average element shape quality as measured by the inverse mean-ratio metric. To solve the resulting large-scale optimization problems, we apply an efficient implementation of an inexact Newton algorithm using the conjugate gradient method with a block Jacobi preconditioner to compute the direction. We show that the block Jacobi preconditioner is positive definite by proving a general theorem concerning the convexity of fractional functions, applying this result to components of the inverse mean-ratio metric, and showing that each block in the preconditioner is invertible. Numerical results obtained with this special-purpose code on several test meshes are presented and used to quantify the impact on solution time and memory requirements of using a modeling language and general-purpose algorithm to solve these problems.

1 Introduction

Discretization methods, such as the finite-element method [8] and the spectral-element method [38], are common techniques for computing an approximate solution to a partial differential equation over a given domain. These methods decompose the domain into a set of elements, triangles or tetrahedra, for example, to produce the mesh used in the approximation scheme. Techniques such as h-refinement, modifying the size of the elements, and p-refinement, modifying the polynomial degree of the elements, are typically applied to improve the accuracy of the approximate solution computed [2]. This paper concerns a supplementary refinement technique, called r-refinement, that modifies the elements to improve their quality [5, 6]. This technique is used in conjunction with h- and p-refinement when solving the partial differential equation.

*This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract W-31-109-Eng-38.

†Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439; e-mail: `tmunson@mcs.anl.gov`

Meshes containing bad quality elements are undesirable because the resulting systems of equations solved can be poorly conditioned, leading to a loss of accuracy, stability, and efficiency [36]. Furthermore, the accuracy in the resulting approximate solution to the partial differential equation can be poor [36]. Improving the element quality according to an appropriate quality metric can have a substantial effect on the total time taken to solve the partial differential equation. For example, in [18] the problem obtained by applying r-refinement and topological modifications took over 25% less time to solve than the original problem.

The use of r-refinement requires both a metric to measure the element quality and a mechanism to modify the elements. Many metrics have been proposed in the literature to measure quality based on quantities such as element size, shape, and smoothness; see [4, 15, 17, 23, 24, 34] and the references therein. This paper concentrates on the inverse mean-ratio metric [26, 24], a shape-quality metric that measures the distance between a trial element and an ideal element, an equilateral triangle, for example. An optimization problem is then solved to minimize the average inverse mean-ratio metric by relocating the vertex positions [16]. Section 2 discusses the inverse mean-ratio metric for two- and three-dimensional elements and the resulting nonlinear optimization problem solved to compute the vertex positions. The objective function for this problem consists of the sum of many fractional functions and is nonconvex. Moreover, the problem can have a large number of variables.

Modeling languages, such as AMPL [14] or GAMS [9], offer convenient environments for specifying and solving optimization problems. These languages are able to generate large problems; gather data from many sources, including databases and spreadsheets; and calculate the derivative information required by the solver. An optimization problem to minimize the average inverse mean-ratio metric for any given mesh can be easily written in a modeling language and solved by applying a general-purpose code, such as LOQO [39, 40] or KNITRO [10, 41].

However, it may be inefficient to incorporate such a model into a code that needs to periodically apply r-refinement either because it is used in conjunction with an adaptive meshing scheme or because the mesh changes with time [3, 37], since the mesh data would have to be written to a file, the modeling environment invoked by a system call, and the results read back into the code from a file. The situation is exacerbated if the code runs in parallel, since no modeling environments operate in parallel. Furthermore, efficiency is required in these applications so that the r-refinement does not become the dominant cost in the overall computation.

Therefore, an efficient special-purpose code to minimize the average inverse mean-ratio metric for any given mesh was developed. In particular, an inexact Newton method [22, 29] was developed to solve these large, nonconvex optimization problems, in which the direction is calculated by applying the conjugate gradient method [32]. A block Jacobi preconditioner is used to reduce the number of conjugate gradient iterations needed to compute the Newton direction. Since the objective function is nonconvex, a proof that the preconditioner is positive definite is required. This proof is provided in Section 3, where a general theorem on the convexity of fractional functions is developed and applied to components of the inverse mean-ratio metric, and each block in the preconditioner matrix is then shown to be invertible.

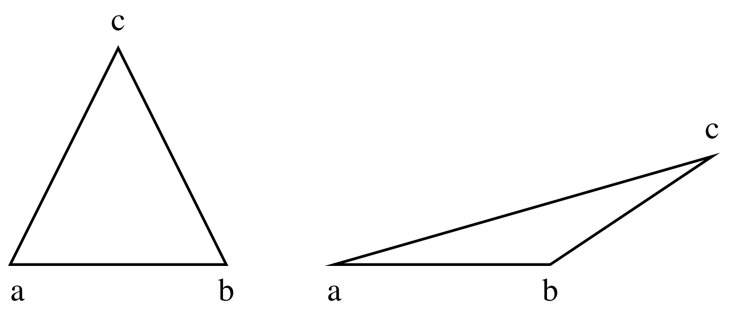

Figure 1: Sample good triangular element (left) and bad triangular element (right) when using the inverse mean-ratio metric referenced to an equilateral triangle.

Translating a model and solver into a special-purpose code and then validating the result is a time-consuming task recommended only if performance is crucial. Tweaking the implementation to make the code more efficient is another critical step requiring a significant time investment. Section 4 discusses the inexact Newton method implemented and the computational techniques used to improve performance. One technique is to reorder the input data to obtain better locality of reference and cache performance [12, 21]. This modification results in a 40% reduction in the overall solution time for the largest meshes tested. The complete inexact Newton code and example meshes are publicly available for download from `http://www.mcs.anl.gov/~tmunson/codes`.

We validated the implementation against equivalent constrained and unconstrained AMPL models. These models and the data files for the test meshes are available for download from `http://www.mcs.anl.gov/~tmunson/models`. Results obtained with the inexact Newton method and those obtained by applying general-purpose optimization codes through the AMPL modeling language are provided in Section 5. In that section we also quantify the price paid for the convenience of using a modeling language on our mesh shape-quality optimization problems in terms of solution time and memory requirements. For the solvers tested on these mesh shape-quality optimization problems, using a general-purpose code within the AMPL modeling language is approximately 9.5 to 100 times slower over the entire set of test meshes and can consume over 100 times the memory. Furthermore, the impact on performance of reordering the mesh is surprising in that the reordering can increase or decrease the number of iterations taken and can affect the success or failure of the algorithms.

2 Shape-Quality Optimization

The two-dimensional version of the inverse mean ratio [26] is a shape-quality metric measuring the distance between a triangular element and an ideal element, an isosceles or equilateral triangle, for example. The description in this section of the inverse mean-ratio metric referenced to an ideal element follows that of Knupp [23, 24] and Freitag and Knupp [15].

Let (a, b, c) be the coordinates for the three vertices in the triangular element shown in Figure 1, where each vertex is an element of $\Re^2$. We define the incidence matrix $A \in \Re^{2\times 2}$

as

$$A := \left[\begin{array}{cc} b-a & c-a \end{array} \right].$$

That is, the incidence matrix is obtained by computing the edges of the element emanating from the first vertex in the coordinate list and concatenating them into a square matrix. Using this definition, we calculate the area of the triangular element as $\frac{1}{2}|\det(A)|$, where $\det(\cdot)$ is the determinant of the square matrix argument. Furthermore, if the element has a nonzero area, then A^{-1} exists.

Let W denote the incidence matrix for an ideal element. A common choice for the ideal element is either an isosceles triangle, where W is the identity matrix, or an equilateral triangle, where

$$W = \left[\begin{array}{cc} 1 & \frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} \end{array} \right].$$

The equilateral ideal element is used for the computational results in Section 5. However, any incidence matrix can be used as long as $\det(W) > 0$.

Assume an arbitrary element with incidence matrix A and an ideal element with incidence matrix W. The quantity AW^{-1} is the identity matrix when the trial element and the ideal element have the same shape and size. If the trial element and the ideal element have the same shape but different sizes, then AW^{-1} is a positive multiple of the identity matrix, where the multiple is the scaling factor.

The inverse mean ratio is then defined as

$$\frac{\|AW^{-1}\|_F^2}{2|\det(AW^{-1})|}.$$

When the trial element and the ideal element have the same shape with a scaling factor of $\sigma > 0$, then the numerator has a value of $2\sigma^2$. This quantity is divided by a term related to the area of the element in order to make the entire measure independent of scaling. Furthermore, the denominator has a value of $2\sigma^2$ when the trial and ideal elements have the same shape. The resulting quantity is a dimensionless measure of the shape of the trial element with respect to the ideal element. The range of the inverse mean ratio is between one and infinity, where a value greater than one means that the trial element and the ideal element have different shapes. This metric is invariant to scaling, translating, and rotating the input values.

A mesh M is defined by a set of vertices V and the elements E that connect these vertices, where each element is an ordered set of three vertices. The set of vertices on the boundary of the mesh is denoted by ∂M; these vertices are fixed for the duration of the computation. We let $x \in \Re^{2\times|V|}$, where $|V|$ is the number of vertices in the mesh, and define

$$A_e(x) = \left[\begin{array}{cc} x_{e_2} - x_{e_1} & x_{e_3} - x_{e_1} \end{array} \right] W^{-1},$$

where $e \in E$ with e_j denoting the jth vertex of element e, and x_i denotes the ith column of the coordinate matrix x. That is, $A_e(x)$ is the incidence matrix for element e times the inverse incidence matrix for the ideal element.

Note that $\det(A_e(x))$ can be a positive or negative quantity for each element depending on how the vertices in the element are labeled; a counterclockwise labeling yields a positive

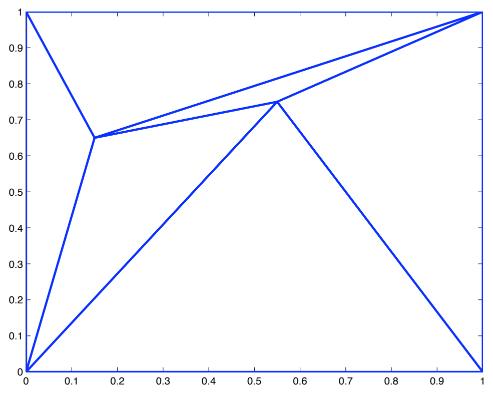

Figure 2: Triangular mesh containing two free vertices with an indefinite Hessian matrix.

determinant, whereas a clockwise labeling yields a negative determinant. A consistent orientation for all of the elements is required for standard discretization methods to work correctly [8]. To enforce a consistent orientation, we impose the constraint that $A_e(x) > 0$ for all $e \in E$.

An optimization problem to minimize the average inverse mean ratio over the entire mesh is then

$$\begin{array}{lll} \min_{x \in \Re^{2 \times |V|}} & \sum_{e \in E} \frac{\|A_e(x)\|_F^2}{2 \det(A_e(x))} & \\ \text{subject to} & \det(A_e(x)) > 0 & \forall e \in E \\ & x_i = \bar{x}_i & \forall i \in \partial M, \end{array}$$

where $\bar{x}_i$ denotes the fixed location of the ith boundary vertex. This optimization problem was also used in [16]. The absolute value in the denominator of the mean-ratio metric has been dropped because the consistent orientation constraints ensure that this quantity is positive.

The objective function is twice continuously differentiable on the open feasible region, but it is not convex. Figure 2 shows a triangular mesh with six elements and two free vertices that uses an equilateral ideal element. The Hessian of the objective function contains one negative eigenvalue with a value of -70 and three positive eigenvalues. Therefore, the objective function is not convex.

Furthermore, the feasible region, the set of points with a consistent orientation, may be neither convex nor connected. However, most of the meshing packages used to construct a mesh for a given domain, such as [33, 35], provide a feasible point. In cases where the starting point is infeasible, an auxiliary optimization problem with a different metric can be solved to calculate a feasible starting point. The problem used to construct a feasible starting point is beyond the scope of this work; see [19], for example.

The inverse mean ratio metric can be generalized to tetrahedral elements. A tetrahedral element consists of an ordered set of four vertices (a, b, c, d), where each vertex is an element of $\Re^3$. The incidence matrix $A \in \Re^{3 \times 3}$ is then defined as

$$A := \left[\begin{array}{ccc} b - a & c - a & d - a \end{array} \right],$$

where the determinant of the incidence matrix is a measure of the volume of the element. As before, let W denote the incidence matrix for the ideal element, and assume that this element has a positive volume so that W^{-1} exists. If the ideal tetrahedral element is an equilateral tetrahedron, then the weight matrix is

$$W = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{6} \\ 0 & 0 & \frac{\sqrt{6}}{3} \end{bmatrix}.$$

An isosceles tetrahedral element is obtained when W is the identity matrix.

The mean ratio for a tetrahedral element with an incidence matrix A referenced to an ideal element with incidence matrix W is then

$$\frac{\|AW^{-1}\|_F^2}{3|\det(AW^{-1})|^{\frac{2}{3}}}.$$

The power in the denominator is necessary in order to make the entire quantity dimensionless and scale invariant. As in the two-dimensional case, the mean ratio metric for tetrahedral elements is invariant to scaling, translating, and rotating the input values.

An optimization problem similar to the two-dimensional case can be constructed for tetrahedral elements. In particular, the objective function uses this form of the inverse mean-ratio metric without the absolute value in the denominator and imposes the consistent orientation constraints $\det(A_e(x)) > 0$ for all $e \in E$.

An assumption made for the shape-quality optimization problem is that the objective function approaches infinity for any sequence of feasible points in which the area of at least one element converges to zero. This assumption is satisfied when the triangular or tetrahedral mesh is edge connected and contains at least two distinct vertices fixed on the boundary of the mesh. A mesh is edge connected if for every pair of elements e^k and e^ℓ there exists an ordered sequence of elements emanating from e^k and terminating with e^ℓ such that all adjacent elements in the sequence share at least two vertices. The two shared vertices form an edge that both elements have in common.

The general argument made to prove this statement is by contradiction. In order for the objective function to be bounded away from infinity, all of the individual element functions must be bounded away from infinity. Therefore, if the area of an element in the sequence of feasible points converges to zero and the inverse mean ratio for that element remains bounded, then the Frobenius norm of the incidence matrix for that element must converge to zero. That is, the length of every edge in the element must converge to zero. Because the mesh is edge connected, at least one edge in all of the neighboring elements must also converge to zero, implying that the areas of all the neighboring elements converge to zero. The argument is then applied inductively to show that the length of all edges in the mesh must converge to zero, a situation that can happen only if all of the vertices converge to a single point. However, two distinct vertices are fixed on the boundary by assumption. Therefore, if the triangular or tetrahedral mesh is edge connected and contains at least two distinct vertices fixed on the boundary, then the objective function approaches infinity for any sequence of feasible points in which the area of at least one element converges to zero.

Therefore, the consistent orientation constraints can be dropped from the optimization problem. In this case, the objective function must return a value of plus infinity whenever the area of one of the elements is nonpositive. Furthermore, a feasible starting point must be supplied to the optimization routine. Once the boundary vertices are fixed and removed, we have an unconstrained optimization problem with an objective function that is twice continuously differentiable on an open set containing the level set. Therefore, a Newton method can be applied to solve this problem.

While the inverse mean-ratio metric is defined for triangular and tetrahedral elements, it can be used with other element types. A quadrilateral element, for example, can be decomposed into four overlapping triangles. In this case, both the trial and ideal quadrilaterals are decomposed into triangles. The inverse mean ratio is then applied to each decomposed triangle in the trial element referenced to the corresponding ideal triangle. The average of these values is a shape-quality metric for the quadrilateral element. Similarly, hexahedral elements can be decomposed into eight overlapping tetrahedra.

3 Preconditioner Properties

We apply an inexact Newton method [22, 29] to the unconstrained optimization problem for minimizing the average inverse mean-ratio metric for a given mesh, where the conjugate gradient method [32] with a block Jacobi preconditioner is used to approximately solve the symmetric system of linear equations. This system of equations can be indefinite because the objective function is not convex. Therefore, we need to prove that the block Jacobi preconditioner is positive definite.

Given a feasible point for the optimization problem, we obtain the block Jacobi preconditioner by taking the Hessian of the objective function, $F(x)$, with respect to each of the vertices. That is,

$$M = \begin{bmatrix} \nabla^2_{x_1,x_1}F(x) & & & & \\ & \ddots & & & \\ & & \nabla^2_{x_i,x_i}F(x) & & \\ & & & \ddots & \\ & & & & \nabla^2_{x_{|V|},x_{|V|}}F(x) \end{bmatrix},$$

where $\nabla^2_{x_i,x_i}F(x) \in \Re^{2\times 2}$ for triangular elements and all off-diagonal blocks are zero. To establish that this matrix is positive definite, we prove that $\nabla^2_{x_i,x_i}F(x)$ is positive definite for each $i = \{1, \ldots, |V|\}$.

This proof is performed in two parts. In Section 3.1, we prove a general theorem about the convexity of fractional functions. This theorem is then applied to components of the inverse mean-ratio metric in Section 3.2 to show that this metric is a convex function in each of the vertices. We then prove that the appropriate parts of the Hessian matrix are invertible. The preconditioner is then shown to be positive definite.

3.1 Convexity Theorem for Fractional Functions

The inverse mean-ratio metric is a nonlinear fractional function $\frac{f}{g}$, where $f : \Re^n \to \Re$ and $g : \Re^n \to \Re$ are nonlinear functions. If f is a nonnegative, convex function and g is a positive, concave function, then $\frac{f}{g}$ is a pseudoconvex function (see [27] Problem 9.6.1, for example). However, pseudoconvexity is not preserved under addition and does not guarantee that the

Hessian matrix is positive semidefinite. Proposition 3.2 provides a set of conditions under which nonlinear fractional functions are proved to be convex.

Definition 3.1 (Uniform Convexity [30]) *Let $f : \Re^n \to \Re$, and let $\Omega \subseteq \Re^n$ be a convex set. The function f is uniformly convex on Ω with constant κ if there exists a constant $\kappa > 0$ such that for all $x \in \Omega$, $y \in \Omega$, and $\lambda \in [0,1]$,*

$$f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y) - \kappa\lambda(1-\lambda)\|y-x\|_2^2.$$

Proposition 3.2 *Let $f : \Re^n \to \Re$ and $g : \Re^n \to \Re$, and let $\Omega \subseteq \Re^n$ be a convex set. Assume the following properties are satisfied:*

1. *g is a positive, concave function on Ω.*
2. *f is a nonnegative, uniformly convex function with constant κ on Ω.*
3. *For all $(x,y) \in \Theta := \left\{(x,y) \in \Omega \times \Omega \mid \frac{f(y)}{g(y)} \geq \frac{f(x)}{g(x)} \text{ and } g(y) \geq g(x)\right\}$,*

$$\left(\frac{f(y)}{g(y)} - \frac{f(x)}{g(x)}\right)(g(y) - g(x)) \leq \kappa\|y-x\|_2^2.$$

Then, $\frac{f}{g}$ is a nonnegative, convex function on Ω.

Proof: The function $\frac{f}{g}$ is nonnegative on Ω, since both g and f are nonnegative functions by Property 1 and Property 2, respectively. We now prove this function is convex on Ω.

If Ω is the empty set, then there is nothing to prove. Therefore, assume Ω is nonempty. Let $x \in \Omega$, $y \in \Omega$, and $\lambda \in [0,1]$ be arbitrary. Define $x(\lambda) = (1-\lambda)x + \lambda y$. Since Ω is a convex set, $x(\lambda) \in \Omega$ and $g(x(\lambda)) > 0$ by Property 1.

By using Property 2 and the fact that $g(x(\lambda)) > 0$, the following inequality is obtained:

$$\frac{f(x(\lambda))}{g(x(\lambda))} \leq (1-\lambda)\frac{f(x)}{g(x(\lambda))} + \lambda\frac{f(y)}{g(x(\lambda))} - \frac{\kappa\lambda(1-\lambda)}{g(x(\lambda))}\|y-x\|_2^2. \tag{1}$$

We obtain an approximation to $\frac{1}{g(x(\lambda))}$ by first evaluating the difference $\frac{1}{g(x(\lambda))} - \frac{1}{g(x)}$ with the following inferences:

$$\frac{1}{g(x(\lambda))} - \frac{1}{g(x)} = \frac{g(x) - g(x(\lambda))}{g(x)g(x(\lambda))} \leq \frac{g(x) - (1-\lambda)g(x) - \lambda g(y)}{g(x)g(x(\lambda))} = \lambda\frac{g(x) - g(y)}{g(x)g(x(\lambda))},$$

where the inequality is a consequence of the concavity and positivity of g on Ω from Property 1. Therefore,

$$\frac{1}{g(x(\lambda))} \leq \frac{1}{g(x)} + \lambda\frac{g(x) - g(y)}{g(x)g(x(\lambda))}. \tag{2}$$

A similar argument shows that

$$\frac{1}{g(x(\lambda))} \leq \frac{1}{g(y)} + (1-\lambda)\frac{g(y) - g(x)}{g(y)g(x(\lambda))}. \tag{3}$$

Equation (1) is then combined with (2) and (3) by using the fact that $f(x) \geq 0$ from Property 2 to obtain

$$\begin{aligned}
\frac{f(x(\lambda))}{g(x(\lambda))} &\leq (1-\lambda)\frac{f(x)}{g(x(\lambda))} + \lambda\frac{f(y)}{g(x(\lambda))} - \frac{\kappa\lambda(1-\lambda)}{g(x(\lambda))}\|y-x\|_2^2 \\
&\leq (1-\lambda)\frac{f(x)}{g(x)} + \lambda\frac{f(y)}{g(y)} + \lambda(1-\lambda)\left(f(x)\frac{g(x)-g(y)}{g(x)g(x(\lambda))} + f(y)\frac{g(y)-g(x)}{g(y)g(x(\lambda))} - \kappa\frac{\|y-x\|_2^2}{g(x(\lambda))}\right) \\
&= (1-\lambda)\frac{f(x)}{g(x)} + \lambda\frac{f(y)}{g(y)} + \frac{\lambda(1-\lambda)}{g(x(\lambda))}\left(\left(\frac{f(y)}{g(y)} - \frac{f(x)}{g(x)}\right)(g(y) - g(x)) - \kappa\|y-x\|_2^2\right).
\end{aligned}$$

Since $\frac{\lambda(1-\lambda)}{g(x(\lambda))}$ is a nonnegative value, we complete the proof by showing

$$\left(\frac{f(y)}{g(y)} - \frac{f(x)}{g(x)}\right)(g(y) - g(x)) - \kappa\|y - x\|_2^2 \leq 0.$$

Without loss of generality, we can assume that x and y have been ordered so that $g(x) \leq g(y)$. Therefore, two cases must be considered.

Case 1 If $g(x) \leq g(y)$ and $\frac{f(x)}{g(x)} > \frac{f(y)}{g(y)}$, then

$$\left(\frac{f(y)}{g(y)} - \frac{f(x)}{g(x)}\right)(g(y) - g(x)) - \kappa\|y - x\|_2^2 \leq 0$$

because the first term is nonpositive.

Case 2 If $g(x) \leq g(y)$ and $\frac{f(x)}{g(x)} \leq \frac{f(y)}{g(y)}$, then $(x, y) \in \Theta$, and Property 3 implies that

$$\left(\frac{f(y)}{g(y)} - \frac{f(x)}{g(x)}\right)(g(y) - g(x)) - \kappa\|y - x\|_2^2 \leq 0.$$

Therefore, for all $x \in \Omega$, $y \in \Omega$, and $\lambda \in [0, 1]$

$$\frac{f((1-\lambda)x+\lambda y)}{g((1-\lambda)x+\lambda y)} \leq (1-\lambda)\frac{f(x)}{g(x)} + \lambda\frac{f(y)}{g(y)},$$

and $\frac{f}{g}$ is a convex function on Ω. $\square$

Property 3 can be interpreted as a joint Lipschitz continuity condition between $\frac{f}{g}$ and g on a restricted set. This condition is necessary when, for example, $f(x) = \|x\|_2^2$ and g is a linear function. In this case, $f(x)$ is a uniformly convex function with $\kappa = 1$. Furthermore, (1), (2), and (3) all become equations in the proof of Proposition 3.2. Since these approximations are now tight, Property 3 must hold with $\kappa = 1$ in order for Case 2 to imply that $\frac{f}{g}$ is a convex function.

3.2 Convexity of the Inverse Mean-Ratio Metric

Recall that the inverse mean-ratio metric is defined as:

$$\frac{\|A(x)\|_F^2}{\det(A(x))^\alpha},$$

where $A(x)$ is the incidence matrix for the trial element times the inverse incidence matrix for the ideal element, W^{-1}, and $\alpha = 1$ for triangles and $\alpha = \frac{2}{3}$ for tetrahedra. All proofs in this section are for triangular elements. The same techniques can be applied for tetrahedral elements; the details can be found in [28].

To fix notation let $0 \leq \alpha \leq 1$; let $(w^a, w^b, w^c) \in \Re^{2\times 3}$ be the coordinates for the ideal element with $\det(W) = \det\left(\left[\begin{array}{cc} w^b - w^a & w^c - w^a \end{array}\right]\right) > 0$; and let $(a, b, c) \in \Re^{2\times 3}$ be the coordinates for the trial element with $\det\left(\left[\begin{array}{cc} b - a & c - a \end{array}\right]\right) > 0$. Moreover, let

$$W^{-1} = \left[\begin{array}{cc} \bar{w}_{1,1} & \bar{w}_{1,2} \\ \bar{w}_{2,1} & \bar{w}_{2,2} \end{array}\right],$$

where $\bar{w}_{i,j}$ is the (i,j) element of W^{-1}.

We then define the following functions:

$$\begin{aligned}
A_a(x) &= \left[\begin{array}{cc} b-x & c-x \end{array}\right] W^{-1} \\
A_b(x) &= \left[\begin{array}{cc} x-a & c-a \end{array}\right] W^{-1} \\
A_c(x) &= \left[\begin{array}{cc} b-a & x-a \end{array}\right] W^{-1}
\end{aligned}$$

$$\begin{aligned}
m_a(x) &= \frac{\|A_a(x)\|_F^2}{\det(A_a(x))^\alpha} \\
m_b(x) &= \frac{\|A_b(x)\|_F^2}{\det(A_b(x))^\alpha} \\
m_c(x) &= \frac{\|A_c(x)\|_F^2}{\det(A_c(x))^\alpha}.
\end{aligned}$$

That is, $A_a(x)$ is the incidence matrix times the inverse incidence matrix for the ideal element as a function of the first vertex position, while $A_b(x)$ and $A_c(x)$ are the corresponding functions for the second and third vertex positions, respectively, while $m_a(x)$, $m_b(x)$, and $m_c(x)$ are the resulting inverse mean-ratio functions. We also define the following sets:

$$\begin{aligned}
\Omega_a &= \left\{x \in \Re^2 \mid \det(A_a(x)) > 0\right\} \\
\Omega_b &= \left\{x \in \Re^2 \mid \det(A_b(x)) > 0\right\} \\
\Omega_c &= \left\{x \in \Re^2 \mid \det(A_c(x)) > 0\right\}.
\end{aligned}$$

The remainder of this section shows that $m_a(x)$, $m_b(x)$, and $m_c(x)$ are each convex functions of x on Ω_a, Ω_b, and Ω_c, respectively, by applying Proposition 3.2. The Hessian matrix is then shown to be positive definite.

Lemma 3.3 demonstrates that the mean ratio is invariant to an even permutation applied to the vertices for both the trial and ideal elements. The permutation needs to be even so that we do not change the sign of $\det(W)$. This invariance means that we need to prove convexity and positive definiteness for only one function since the others can be obtained by applying a permutation.

Lemma 3.3 *Let $(w^a, w^b, w^c) \in \Re^{2\times 3}$ be given such that $\det(W) > 0$, and let $(a,b,c) \in \Re^{2\times 3}$ be arbitrary. Then, the following are equivalent:*

1. $\left[\begin{array}{cc} b-a & c-a \end{array}\right]\left[\begin{array}{cc} w^b-w^a & w^c-w^a \end{array}\right]^{-1}$
2. $\left[\begin{array}{cc} c-b & a-b \end{array}\right]\left[\begin{array}{cc} w^c-w^b & w^a-w^b \end{array}\right]^{-1}$.

Proof:

$$\begin{aligned}
&\left[\begin{array}{cc} b-a & c-a \end{array}\right]\left[\begin{array}{cc} w^b-w^a & w^c-w^a \end{array}\right]^{-1} \\
&\quad = \left[\begin{array}{cc} c-b & a-b \end{array}\right]\left[\begin{array}{cc} 0 & 1 \\ -1 & -1 \end{array}\right]\left(\left[\begin{array}{cc} w^c-w^b & w^a-w^b \end{array}\right]\left[\begin{array}{cc} 0 & 1 \\ -1 & -1 \end{array}\right]\right)^{-1} \\
&\quad = \left[\begin{array}{cc} c-b & a-b \end{array}\right]\left[\begin{array}{cc} 0 & 1 \\ -1 & -1 \end{array}\right]\left[\begin{array}{cc} 0 & 1 \\ -1 & -1 \end{array}\right]^{-1}\left[\begin{array}{cc} w^c-w^b & w^a-w^b \end{array}\right]^{-1} \\
&\quad = \left[\begin{array}{cc} c-b & a-b \end{array}\right]\left[\begin{array}{cc} w^c-w^b & w^a-w^b \end{array}\right]^{-1}.
\end{aligned}$$

□

Lemma 3.4 *For any weight matrix W^{-1},*

1. $\det(A_c(x))$ *is a linear function of* x.
2. Ω_c *is a convex set.*

Proof: From the properties of the determinant,

$$\begin{aligned}\det(A_c(x)) &= \det\left(\begin{bmatrix} b_1-a_1 & x_1-a_1 \\ b_2-a_2 & x_2-a_2\end{bmatrix} W^{-1}\right) \\ &= \det\left(\begin{bmatrix} b_1-a_1 & x_1-a_1 \\ b_2-a_2 & x_2-a_2\end{bmatrix}\right)\det\left(W^{-1}\right) \\ &= \left((b_1-a_1)(x_2-a_2)-(b_2-a_2)(x_1-a_1)\right)\det\left(W^{-1}\right)\end{aligned}$$

This calculation shows that $\det(A_c(x))$ is a linear function of x. Furthermore, Ω_c consists of a strict linear inequality, which forms a convex set. □

Lemma 3.5 *For any weight matrix W^{-1} and for any $0 \le \alpha \le 1$, $\det(A_c(x))^\alpha$ is a concave function of x on Ω_c.*

Proof: Since $\det(A_c(x))$ is a linear function of x by Lemma 3.4,

$$\det(A_c((1-\lambda)x+\lambda y)) = (1-\lambda)\det(A_c(x)) + \lambda\det(A_c(y))$$

for any $\lambda \in [0,1]$. If $x \in \Omega_c$ and $y \in \Omega_c$, then $(1-\lambda)x+\lambda y \in \Omega_c$ because Ω_c is a convex set by Lemma 3.4. Therefore, $\det(A_c((1-\lambda)x+\lambda y)) > 0$. The power is applied to both sides of the equation to obtain

$$\begin{aligned}\det(A_c((1-\lambda)x+\lambda y))^\alpha &= ((1-\lambda)\det(A_c(x))+\lambda\det(A_c(y)))^\alpha \\ &\ge (1-\lambda)\det(A_c(x))^\alpha + \lambda\det(A_c(y))^\alpha,\end{aligned}$$

where the last inequality holds because ψ^α is a concave function on the region $\psi \ge 0$ for any $0 \le \alpha \le 1$ [31]. □

Lemma 3.6 *For any weight matrix W^{-1} with $\det(W^{-1}) > 0$, $\|A_c(x)\|_F^2$ is a uniformly convex function of x with constant $\kappa = \bar{w}_{2,1}^2 + \bar{w}_{2,2}^2 > 0$.*

Proof: The Hessian matrix for $\|A_c(x)\|_F$ is $2(\bar{w}_{2,1}^2 + \bar{w}_{2,2}^2)I$, where I is the identity matrix. Therefore, this matrix is uniformly positive definite with constant $2(\bar{w}_{2,1}^2 + \bar{w}_{2,2}^2)$. The relationship between equivalent definitions of uniform convexity then imply that

$$\|A_c((1-\lambda)x+\lambda y)\|_F^2 \le (1-\lambda)\|A_c(x)\|_F^2 + \lambda\|A_c(y)\|_F^2 - \left(\bar{w}_{2,1}^2 + \bar{w}_{2,2}^2\right)\lambda(1-\lambda)\|y-x\|_2^2.$$

Since $\det(W^{-1}) > 0$, either $\bar{w}_{2,1} \ne 0$ or $\bar{w}_{2,2} \ne 0$. Hence, $\|A_c(x)\|_F^2$ is a uniformly convex function with $\kappa = \bar{w}_{2,1}^2 + \bar{w}_{2,2}^2 > 0$. □

Lemma 3.7 *Let W^{-1} be any weight matrix with $\det(W^{-1}) > 0$, and let $0 \le \alpha \le 1$. Define $\Omega = \Omega_c$, $f(x) = \|A_c(x)\|_F^2$, and $g(x) = \det(A_c(x))^\alpha$. Then for any $(x, y) \in \Theta$, where Θ is defined in Proposition 3.2,*

$$\left(\frac{f(y)}{g(y)} - \frac{f(x)}{g(x)}\right)(g(y) - g(x)) \le (\bar{w}_{2,1}^2 + \bar{w}_{2,2}^2)\|y - x\|_2^2.$$

Proof: This lemma is proved by showing that

$$\sup_{(x,y)\in\Theta} \left(\frac{f(y)}{g(y)} - \frac{f(x)}{g(x)}\right)(g(y) - g(x)) - (\bar{w}_{2,1}^2 + \bar{w}_{2,2}^2)\|y - x\|_2^2 \le 0.$$

If Θ is the empty set, then there is nothing to prove. Therefore, assume Θ is nonempty, and let $(x, y) \in \Theta$. Since $\det(A_c(x)) > 0$, $\|b - a\|_2 > 0$. We then make the change of variables $x = R\bar{x} + a$ and $y = R\bar{y} + a$, where R is an orthogonal matrix with $\det(R) = 1$ defined as

$$R = \begin{bmatrix} \frac{b_1 - a_1}{\|b-a\|_2} & \frac{a_2 - b_2}{\|b-a\|_2} \\ \frac{b_2 - a_2}{\|b-a\|_2} & \frac{b_1 - a_1}{\|b-a\|_2} \end{bmatrix}.$$

We use the following definitions throughout the remainder of this section.

$$\begin{aligned} d &= \|b - a\|_2 \\ \nu &= d\det(W^{-1}) \\ \Delta &= \bar{w}_{2,1}^2 + \bar{w}_{2,2}^2 \\ \delta(\xi) &= (d\bar{w}_{1,1} + \bar{w}_{2,1}\xi)^2 + (d\bar{w}_{1,2} + \bar{w}_{2,2}\xi)^2 \end{aligned}$$

Note that $\Delta > 0$ by Lemma 3.5 and $\nu > 0$.

We now have the following expressions for $\|A_c(x)\|_F^2$, $\det(A_c(x))$, and $\|y - x\|_2^2$:

$$\begin{aligned} \|A_c(x)\|_F^2 &= \|A_c(R\bar{x} + b)\|_F^2 \\ &= \left\|\begin{bmatrix} b - a & R\bar{x} + a - a \end{bmatrix} W^{-1}\right\|_F^2 \\ &= \left\|R\begin{bmatrix} R^T(b - a) & \bar{x} \end{bmatrix} W^{-1}\right\|_F^2 \\ &= \left\|\begin{bmatrix} d & \bar{x}_1 \\ 0 & \bar{x}_2 \end{bmatrix}\begin{bmatrix} \bar{w}_{1,1} & \bar{w}_{1,2} \\ \bar{w}_{2,1} & \bar{w}_{2,2} \end{bmatrix}\right\|_F^2 \\ &= \delta(\bar{x}_1) + \Delta\bar{x}_2^2 \\ \det(A(x)) &= \det\left(R\begin{bmatrix} d & \bar{x}_1 \\ 0 & \bar{x}_2 \end{bmatrix} W^{-1}\right) \\ &= \nu\bar{x}_2 \end{aligned}$$

$$\begin{aligned} \|y - x\|_2^2 &= \|R\bar{y} + a - R\bar{x} - a\|_2^2 \\ &= \|R(\bar{y} - \bar{x})\|_2^2 \\ &= \|\bar{y} - \bar{x}\|_2^2, \end{aligned}$$

where the orthogonality of R is used in the norm calculations and $\det(R) = 1$ is used in the determinant. The optimization problem we want to solve is then

$$\begin{array}{ll} \sup_{\bar{x}\in\Re^2, \bar{y}\in\Re^2} & \left(\frac{\delta(\bar{y}_1) + \Delta\bar{y}_2^2}{(\nu\bar{y}_2)^\alpha} - \frac{\delta(\bar{x}_1) + \Delta\bar{x}_2^2}{(\nu\bar{x}_2)^\alpha}\right)((\nu\bar{y}_2)^\alpha - (\nu\bar{x}_2)^\alpha) - \Delta\|\bar{y} - \bar{x}\|_2^2 \\ \text{subject to} & \nu\bar{y}_2 \ge \nu\bar{x}_2 > 0 \\ & (\nu\bar{y}_2)^\alpha \ge (\nu\bar{x}_2)^\alpha \\ & \frac{\delta(\bar{y}_1) + \Delta\bar{y}_2^2}{(\nu\bar{y}_2)^\alpha} \ge \frac{\delta(\bar{x}_1) + \Delta\bar{x}_2^2}{(\nu\bar{x}_2)^\alpha}. \end{array}$$

Eliminating ν from the problem because it is a positive constant, and dropping the last two constraints, we obtain the following optimization problem, which provides an upper bound on the supremum:

$$\begin{array}{ll}\sup_{\bar{x}\in\Re^2,\bar{y}\in\Re^2} & \left(\frac{\delta(\bar{y}_1)+\Delta\bar{y}_2^2}{\bar{y}_2^\alpha}-\frac{\delta(\bar{x}_1)+\Delta\bar{x}_2^2}{\bar{x}_2^\alpha}\right)(\bar{y}_2^\alpha-\bar{x}_2^\alpha)-\Delta\|\bar{y}-\bar{x}\|_2^2 \\ \text{subject to} & \bar{y}_2\geq\bar{x}_2>0.\end{array}$$

Examining those terms involving $\bar{y}_2^2$ and $\bar{x}_2^2$, we obtain

$$\begin{aligned}\Delta((\bar{y}_2^{2-\alpha}-\bar{x}_2^{2-\alpha})(\bar{y}_2^\alpha-\bar{x}_2^\alpha)-(\bar{y}_2-\bar{x}_2)^2) &= \Delta(\bar{y}_2^2-\bar{y}_2^{2-\alpha}\bar{x}_2^\alpha-\bar{y}_2^\alpha\bar{x}_2^{2-\alpha}+\bar{x}_2^2-(\bar{y}_2-\bar{x}_2)^2) \\ &= \Delta(2\bar{y}_2\bar{x}_2-\bar{y}_2^{2-\alpha}\bar{x}_2^\alpha-\bar{y}_2^\alpha\bar{x}_2^{2-\alpha}) \\ &= \Delta\bar{y}_2\bar{x}_2\left(2-\frac{\bar{y}_2^{1-\alpha}}{\bar{x}_2^{1-\alpha}}-\frac{\bar{x}_2^{1-\alpha}}{\bar{y}_2^{1-\alpha}}\right) \\ &\leq 0,\end{aligned}$$

where the last inequality is obtained from the arithmetic-geometric mean inequality. After removing these terms, we are left with the following optimization problem, which provides an upper bound on the supremum:

$$\begin{array}{ll}\sup_{\bar{x}\in\Re^2,\bar{y}\in\Re^2} & \left(\frac{\delta(\bar{y}_1)}{\bar{y}_2^\alpha}-\frac{\delta(\bar{x}_1)}{\bar{x}_2^\alpha}\right)(\bar{y}_2^\alpha-\bar{x}_2^\alpha)-\Delta(\bar{y}_1-\bar{x}_1)^2 \\ \text{subject to} & \bar{y}_2\geq\bar{x}_2>0.\end{array}$$

We now write $\bar{y}_2^\alpha=\beta\bar{x}_2^\alpha$ for $\beta\geq 1$ because $\bar{x}_2^\alpha>0$, eliminate $\bar{y}_2$ and $\bar{x}_2$, and rearrange the terms to obtain the equivalent optimization problem:

$$\sup_{\bar{y}_1\in\Re,\beta\geq 1}\sup_{\bar{x}_1\in\Re}\ \frac{1}{\beta}\left((\beta-1)\delta(\bar{y}_1)-\beta(\beta-1)\delta(\bar{x}_1)-\beta\Delta(\bar{y}_1-\bar{x}_1)^2\right).$$

Note that the objective function is strongly concave in the $\bar{x}_1$ variable. Therefore, we can set to zero the gradient of the objective function with respect to $\bar{x}_1$ to derive the optimal solution for $\bar{x}_1$ given $\bar{y}_1$ and β.

$$\begin{aligned}\nabla_{\bar{x}_1}\text{obj}(\bar{x}_1,\bar{y}_1,\beta) &= -2((\beta-1)((d\bar{w}_{1,1}+\bar{w}_{2,1}\bar{x}_1)\bar{w}_{2,1}+(d\bar{w}_{1,2}+\bar{w}_{2,2}\bar{x}_1)\bar{w}_{2,2}+\Delta(\bar{x}_1-\bar{y}_1)) \\ &= -2((\beta-1)(\Delta\bar{x}_1+d(\bar{w}_{1,1}\bar{w}_{2,1}+\bar{w}_{1,2}\bar{w}_{2,2})+\Delta(\bar{x}_1-\bar{y}_1)) \\ &= -2(\beta\Delta\bar{x}_1+(\beta-1)d(\bar{w}_{1,1}\bar{w}_{2,1}+\bar{w}_{1,2}\bar{w}_{2,2})-\Delta\bar{y}_1) \\ &= -2\beta\Delta\left(\bar{x}_1+d\frac{\beta-1}{\beta}\left(\frac{\bar{w}_{1,1}\bar{w}_{2,1}+\bar{w}_{1,2}\bar{w}_{2,2}}{\Delta}\right)-\frac{\bar{y}_1}{\beta}\right).\end{aligned}$$

Therefore,

$$\bar{x}_1=\frac{\bar{y}_1}{\beta}-d\frac{\beta-1}{\beta}\left(\frac{\bar{w}_{1,1}\bar{w}_{2,1}+\bar{w}_{1,2}\bar{w}_{2,2}}{\Delta}\right).$$

Substituting this quantity into the objective function gives an equivalent problem

$$\sup_{\bar{y}_1\in\Re,\beta\geq 1}\ \frac{1}{\beta}\left(\begin{array}{c}(\beta-1)\delta(\bar{y}_1)-\beta(\beta-1)\delta\left(\frac{\bar{y}_1}{\beta}-d\frac{\beta-1}{\beta}\left(\frac{\bar{w}_{1,1}\bar{w}_{2,1}+\bar{w}_{1,2}\bar{w}_{2,2}}{\Delta}\right)\right)- \\ \beta\Delta\left(\bar{y}_1-\frac{\bar{y}_1}{\beta}+d\frac{\beta-1}{\beta}\left(\frac{\bar{w}_{1,1}\bar{w}_{2,1}+\bar{w}_{1,2}\bar{w}_{2,2}}{\Delta}\right)\right)^2\end{array}\right),$$

which simplifies to

$$\sup_{\beta\geq 1}-\frac{(\beta-1)^2}{\beta}\left(\frac{\nu^2}{\Delta}\right).$$

The constant in this optimization problem is positive since $\nu>0$ and $\Delta>0$. Hence, the superemum is zero. □

Lemma 3.8 *Let W^{-1} be any weight matrix with* $\det(W^{-1}) > 0$*, and let* $0 \le \alpha \le 1$*. Then for any $x \in \Omega_c$, $\nabla^2_{x,x} m_c(x)$ is invertible.*

Proof: If Ω_c is empty, then there is nothing to prove. Therefore, let Ω_c be nonempty, and let $x \in \Omega_c$. Define R to be as in the proof of Lemma 3.7. Then we have

$$m_c(x) = m_c(R\bar{x} + a)$$

where $\bar{x} = R^T(x - a)$. By the chain rule,

$$\nabla_x m_c(x) = [\nabla_{\bar{x}} m_c(R\bar{x} + a)]\, R^T$$

and

$$\nabla^2_{x,x} m_c(x) = R \left[\nabla^2_{\bar{x},\bar{x}} m_c(R\bar{x} + a)\right] R^T.$$

We can ignore the terms involving R, since R is an orthogonal matrix with $\det(R) = 1$.

Using the definitions from Lemma 3.7 we have by direct computation that

$$\nabla^2_{\bar{x},\bar{x}} m_c(R\bar{x} + a) = \begin{bmatrix} \frac{2\Delta}{\nu^\alpha \bar{x}_2^\alpha} & -\frac{2\alpha(\Delta\bar{x}_1 + d(\bar{w}_{1,1}\bar{w}_{2,1} + \bar{w}_{1,2}\bar{w}_{2,2}))}{\nu^\alpha \bar{x}_2^{\alpha+1}} \\ -\frac{2\alpha(\Delta\bar{x}_1 + d(\bar{w}_{1,1}\bar{w}_{2,1} + \bar{w}_{1,2}\bar{w}_{2,2}))}{\nu^\alpha \bar{x}_2^{\alpha+1}} & \frac{(2-\alpha)(1-\alpha)\Delta}{\nu^\alpha \bar{x}_2^\alpha} + \frac{\alpha(\alpha+1)\delta(\bar{x}_1)}{\nu^\alpha \bar{x}_2^{\alpha+2}} \end{bmatrix}$$

Rewriting, we obtain the following matrix:

$$\frac{1}{\nu^\alpha \bar{x}_2^{\alpha+2}} \begin{bmatrix} 2\Delta\bar{x}_2^2 & -2\alpha(\Delta\bar{x}_1 + d(\bar{w}_{1,1}\bar{w}_{2,1} + \bar{w}_{1,2}\bar{w}_{2,2}))\bar{x}_2 \\ -2\alpha(\Delta\bar{x}_1 + d(\bar{w}_{1,1}\bar{w}_{2,1} + \bar{w}_{1,2}\bar{w}_{2,2}))\bar{x}_2 & (2-\alpha)(1-\alpha)\Delta\bar{x}_2^2 + \alpha(\alpha+1)\delta(\bar{x}_1) \end{bmatrix}.$$

Since $\bar{x}_2 > 0$ because $x \in \Omega_c$ and $\nu > 0$, we can ignore the positive coefficient. The determinant of this matrix is then

$$2(2-\alpha)(1-\alpha)\Delta^2\bar{x}_2^4 + 2\alpha(\alpha+1)\delta(\bar{x}_1)\Delta\bar{x}_2^2 - 4\alpha^2(\Delta\bar{x}_1 + d(\bar{w}_{1,1}\bar{w}_{2,1} + \bar{w}_{1,2}\bar{w}_{2,2}))^2\bar{x}_2^2.$$

The first term is nonnegative and strictly positive whenever $0 \le \alpha < 1$ since $\bar{x}_2 > 0$.

We now concentrate on the last two terms:

$$\begin{aligned}
& 2\alpha(\alpha+1)\delta(\bar{x}_1)\Delta\bar{x}_2^2 - 4\alpha^2(\Delta\bar{x}_1 + d(\bar{w}_{1,1}\bar{w}_{2,1} + \bar{w}_{1,2}\bar{w}_{2,2}))^2\bar{x}_2^2 \\
= \; & 2\alpha\bar{x}_2^2 \left(\begin{array}{l} (\alpha+1)((d\bar{w}_{1,1} + \bar{w}_{2,1}\bar{x}_1)^2 + (d\bar{w}_{1,2} + \bar{w}_{2,2}\bar{x}_1)^2)(\bar{w}_{2,1}^2 + \bar{w}_{2,2}^2) - \\ 2\alpha \left(\begin{array}{l} (d\bar{w}_{1,1} + \bar{w}_{2,1}\bar{x}_1)^2\bar{w}_{2,1}^2 + 2(d\bar{w}_{1,1} + \bar{w}_{2,1}\bar{x}_1)(d\bar{w}_{1,2} + \bar{w}_{2,2}\bar{x}_1)\bar{w}_{2,1}\bar{w}_{2,2} + \\ (d\bar{w}_{1,2} + \bar{w}_{2,2}\bar{x}_1)^2\bar{w}_{2,2}^2 \end{array} \right) \end{array} \right) \\
= \; & 2\alpha\bar{x}_2^2 \left(\begin{array}{l} (1+\alpha)((d\bar{w}_{1,1} + \bar{w}_{2,1}\bar{x}_1)^2\bar{w}_{2,2}^2 + (d\bar{w}_{1,2} + \bar{w}_{2,2}\bar{x}_1)^2\bar{w}_{2,1}^2) + \\ (1-\alpha)((d\bar{w}_{1,1} + \bar{w}_{2,1}\bar{x}_1)^2\bar{w}_{2,1}^2 + (d\bar{w}_{1,2} + \bar{w}_{2,2}\bar{x}_1)^2\bar{w}_{2,2}^2) - \\ 4\alpha(d\bar{w}_{1,1} + \bar{w}_{2,1}\bar{x}_1)(d\bar{w}_{1,2} + \bar{w}_{2,2}\bar{x}_1)\bar{w}_{2,1}\bar{w}_{2,2} \end{array} \right) \\
= \; & 2\alpha\bar{x}_2^2 \left(\begin{array}{l} (1+\alpha)((d\bar{w}_{1,1} + \bar{w}_{2,1}\bar{x}_1)\bar{w}_{2,2} - (d\bar{w}_{1,2} + \bar{w}_{2,2}\bar{x}_1)\bar{w}_{2,1})^2 + \\ (1-\alpha)((d\bar{w}_{1,1} + \bar{w}_{2,1}\bar{x}_1)^2\bar{w}_{2,1}^2 + (d\bar{w}_{1,2} + \bar{w}_{2,2}\bar{x}_1)^2\bar{w}_{2,2}^2) + \\ 2(1-\alpha)(d\bar{w}_{1,1} + \bar{w}_{2,1}\bar{x}_1)(d\bar{w}_{1,2} + \bar{w}_{2,2}\bar{x}_1)\bar{w}_{2,1}\bar{w}_{2,2} \end{array} \right) \\
= \; & 2\alpha\bar{x}_2^2 \left(\begin{array}{l} (1+\alpha)((d\bar{w}_{1,1} + \bar{w}_{2,1}\bar{x}_1)\bar{w}_{2,2} - (d\bar{w}_{1,2} + \bar{w}_{2,2}\bar{x}_1)\bar{w}_{2,1})^2 + \\ (1-\alpha)((d\bar{w}_{1,1} + \bar{w}_{2,1}\bar{x}_1)\bar{w}_{2,1} + (d\bar{w}_{1,2} + \bar{w}_{2,2}\bar{x}_1)\bar{w}_{2,2})^2 \end{array} \right) \\
= \; & 2\alpha\bar{x}_2^2((1+\alpha)\nu^2 + (1-\alpha)((d\bar{w}_{1,1} + \bar{w}_{2,1}\bar{x}_1)\bar{w}_{2,1} + (d\bar{w}_{1,2} + \bar{w}_{2,2}\bar{x}_1)\bar{w}_{2,2})^2).
\end{aligned}$$

Since $\nu > 0$, this quantity is nonnegative and positive when $0 < \alpha \le 1$.

Combining these two conditions, we have that the determinant is positive for any given $0 \le \alpha \le 1$. Hence, $\nabla^2_{x,x} m_c(x)$ is invertible for any $x \in \Omega_c$. □

Theorem 3.9 *Let W^{-1} be any weight matrix with $\det(W^{-1}) > 0$, and let $0 \leq \alpha \leq 1$. Then, $m_c(x)$ is a nonnegative, convex function of x on Ω_c. Furthermore, for any $x \in \Omega_c$, $\nabla^2_{x,x} m_c(x)$ is positive definite.*

Proof: Lemma 3.4 shows that Ω_c is a convex set, Lemma 3.5 demonstrates that $\det(A_c(x))^\alpha$ is a concave function on Ω_c, and $\det(A_c(x))^\alpha > 0$ for any $x \in \Omega_c$. Therefore, Property 1 of Proposition 3.2 is satisfied. Furthermore, $\|A_c(x)\|_F^2$ is a nonnegative function of x and Lemma 3.6 shows that $\|A_c(x)\|_F^2$ is uniformly convex with constant $\bar{w}_{2,1}^2 + \bar{w}_{2,2}^2 > 0$. Therefore, Property 2 of Proposition 3.2 holds. Lemma 3.7 shows that Property 3 is satisfied. Therefore, by Proposition 3.2, $m_c(x)$ is a nonnegative, convex function of x on Ω_c.

Since $m_c(x)$ is convex and twice continuously differentiable on Ω_c, $\nabla^2_{x,x} m_c(x)$ is positive semidefinite for any $x \in \Omega_c$. However, Lemma 3.8 shows that the Hessian matrix is also invertible for any $x \in \Omega_c$. Therefore, we conclude that $\nabla^2_{x,x} m_c(x)$ is positive definite for any $x \in \Omega_c$. $\Box$

Corollary 3.10 *Let W^{-1} be any weight matrix with $\det(W^{-1}) > 0$, and let $0 \leq \alpha \leq 1$. Then $\nabla^2_{x,x} m_a(x)$ is positive definite for any $x \in \Omega_a$ and $\nabla^2_{x,x} m_b(x)$ is positive definite for any $x \in \Omega_b$.*

Proof: By Lemma 3.3,

$$\begin{aligned} A_a(x) &= \begin{bmatrix} b - x & c - x \end{bmatrix} \begin{bmatrix} w^b - w^a & w^c - w^a \end{bmatrix}^{-1} \\ &= \begin{bmatrix} c - b & x - b \end{bmatrix} \begin{bmatrix} w^c - w^b & w^a - w^b \end{bmatrix}^{-1}. \end{aligned}$$

We now apply Theorem 3.9 to $m_a(x)$ using this equivalent redefinition. Therefore, $\nabla^2_{x,x} m_a(x)$ is positive definite for any $x \in \Omega_a$. A similar argument using two applications of Lemma 3.3 shows that $\nabla^2_{x,x} m_b(x)$ is positive definite for any $x \in \Omega_b$. $\Box$

Corollary 3.11 *Let W^{-1} be any weight matrix with $\det(W^{-1}) > 0$, and let $x \in \Re^{2\times|V|}$ be given such that $\det(A_e(x)) > 0$ for all $e \in E$. Then, the block Jacobi preconditioner for the shape-quality optimization problem using the mean-ratio metric is positive definite.*

Proof: The objective function, $F(x)$, for the shape-quality optimization problem consists of the sum of the mean-ratio metric for each element. Therefore,

$$\nabla^2_{x_i,x_i} F(x) = \sum_{\{e \in E | e_1 = i\}} \nabla^2_{x_i,x_i} m_a(x_i) + \sum_{\{e \in E | e_2 = i\}} \nabla^2_{x_i,x_i} m_b(x_i) + \sum_{\{e \in E | e_3 = i\}} \nabla^2_{x_i,x_i} m_c(x_i),$$

where $a = x_{e_1}$, $b = x_{e_2}$, and $c = x_{e_3}$ in the mean ratio metric for each $e \in E$. Theorem 3.9 and Corollary 3.10 now imply that $\nabla^2_{x_i,x_i} F(x)$ is positive definite because the sum of positive definite matrices is also positive definite. Therefore, the block Jacobi preconditioner is positive definite. $\Box$

We have proved that the inverse mean-ratio metric for triangular and tetrahedral elements is a convex function of each coordinate, with the diagonal components of the Hessian matrix being positive definite for any $0 \leq \alpha \leq 1$. In particular, we have an edge-length metric when $\alpha = 0$ and a generalized form of the inverse mean ratio when $0 < \alpha \leq 1$. Note that when $\alpha = 0$, the argument made for removing the consistent orientation constraints in the optimization problem is no longer valid. Furthermore, while we have assumed that $\det(W^{-1}) > 0$, the proofs work with some modification when $\det(W^{-1}) < 0$.

4 Algorithm and Implementation

An inexact Newton method [22, 29] with an Armijo linesearch [1] was coded to solve the mesh shape-quality optimization problem. In particular, given x^k, the algorithm computes a direction d^k by solving the symmetric system of linear equations

$$\nabla^2 F(x^k) d^k = -\nabla F(x^k)$$

by applying a conjugate gradient method with a block Jacobi preconditioner [32]. Since the Hessian can be indefinite, the conjugate gradient method may terminate with a direction of negative curvature. In such a case, the base of the direction is used as the starting point for the linesearch. If

$$\nabla F(x^k)^T d^k \geq -\rho \|d^k\|_2^p,$$

where $\rho > 0$ and $p \geq 2$ are constants, then the steepest descent direction $-\nabla F(x^k)$ is used. The Armijo linesearch finds the smallest nonnegative integer m such that

$$F(x^k + \beta^m d^k) \leq F(x^k) + \sigma \beta^m \nabla F(x^k)^T d^k,$$

where $0 < \sigma < \frac{1}{2}$ and $0 < \beta < 1$ are constants. The iterate is then updated with the rule $x^{k+1} = x^k + \beta^m d^k$, and a new direction is computed. The algorithm terminates when $\|\nabla F(x^k)\|_2$ is less than 1.0×10^{-6}.

The average inverse mean-ratio objective function requires that a value of plus infinity be returned whenever the consistent orientation conditions are not satisfied. Therefore, if the area of at least one element is smaller than $\epsilon = 1.0 \times 10^{-14}$, we consider the consistent orientation conditions to be violated, and the objective function is set to plus infinity in the linesearch.

The gradient and Hessian of the objective function are calculated analytically by assembling the gradients and Hessians for each element function into a vector and symmetric sparse matrix. Only the upper triangular part of the Hessian matrix is stored in a block compressed sparse row format. Each block of the Hessian matrix corresponds to a vertex-vertex interaction in the original mesh. In order to facilitate the assembly of the Hessian matrix, once the sparsity pattern is obtained, an additional vector is calculated that tells the offset into the Hessian matrix data vector where the element Hessians are to be accumulated. The gradient and Hessian of the elements with respect to vertices fixed on the boundary of the mesh are ignored.

The code for calculating the gradient of the element function uses the reverse mode of automatic differentiation [7, 20]. The code was written and refined by hand to eliminate unnecessary operations, resulting in a more efficient gradient evaluation. The Hessian calculation uses the forward mode of differentiation on the gradient evaluation. The resulting code was written using matrix-matrix products for efficiency. An evaluation routine that computes only the gradient of the element has also been coded that requires fewer floating-point operations than the function plus gradient evaluation. A similar routine has been provided to compute only the Hessian evaluation.

The conjugate gradient method terminates if the system of equations is solved to within a specified tolerance, if a direction of negative curvature is encountered, or if 100 conjugate gradient iterations have been performed. The conjugate gradient implementation terminates when

$$\|\nabla^2 F(x^k)d^k + \nabla F(x^k)\|_2 \leq 10^{-2} \times \|\nabla F(x^k)\|_2.$$

That is, the relative tolerance is used for the termination test. No steepest descent directions were used in the tests performed in Section 5. Therefore, the code simply checks that the resulting direction is a descent direction, $\nabla F(x^k)^T d^k < 0$, and terminates with an error message if this condition is not satisfied.

The block Jacobi preconditioner uses the 2×2 or 3×3 matrices on the diagonal of the Hessian matrix depending on whether the problem is triangular or tetrahedral, respectively. An LDL^T factorization of each diagonal matrix is performed when calculating the preconditioner, where L has ones on the diagonal. When the preconditioner is applied, $y = L^{-T}(D^{-1}(L^{-1}x))$ is calculated. In order to eliminate all division operations when the preconditioner is applied, D^{-1} is stored instead of D. Each diagonal block of the Hessian matrix is positive definite by Corollary 3.11, even though the overall Hessian matrix is indefinite in general. Therefore, no checks for indefiniteness are performed during the computation of the preconditioner.

The Armijo linesearch is slightly modified from the literature; the full step length is taken if the gradient of the objective function is within 100 times the convergence tolerance, regardless of the objective function value. This rule is used because the objective function may increase slightly as a result of roundoff errors in the calculation of the objective function, although the algorithm is close to a solution. In order to implement this test, a function plus gradient evaluation is performed for the full step, and only function evaluations are performed during the remainder of the linesearch. If the full step is rejected, a gradient evaluation is performed at the accepted step for use in the termination test.

In order to obtain good locality of reference, the vertices and elements in the initial mesh are reordered by using a breadth-first search ordering [32] prior to applying the inexact Newton method. The ordering starts by selecting the (boundary) vertex farthest from the origin as a starting point. A breadth-first search of the vertices in the mesh is then performed. The order in which the vertices were visited is reversed, as in the reverse Cuthill-McKee ordering [11], to obtain a symmetric permutation of the vertices for the optimization problem. The elements are then ordered according to when they are referenced by the vertices. Other orderings can be applied that may give rise to further improvements in performance [21].

The source code is publicly available at `http://www.mcs.anl.gov/~tmunson/codes`.

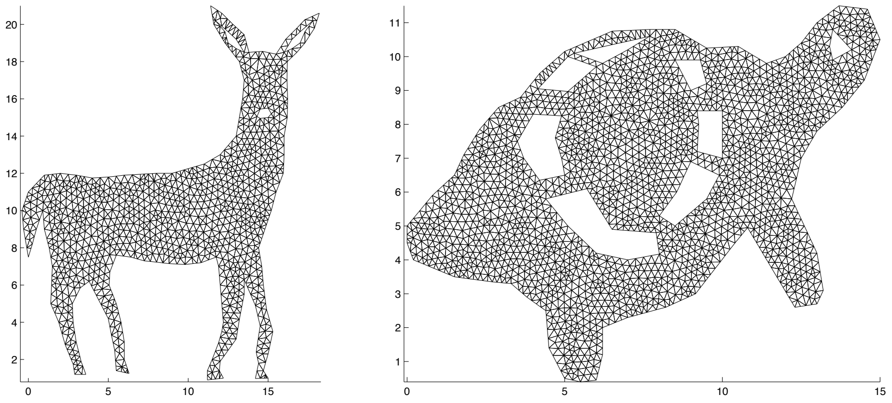

Figure 3: Smoothed deer and turtle triangular meshes.

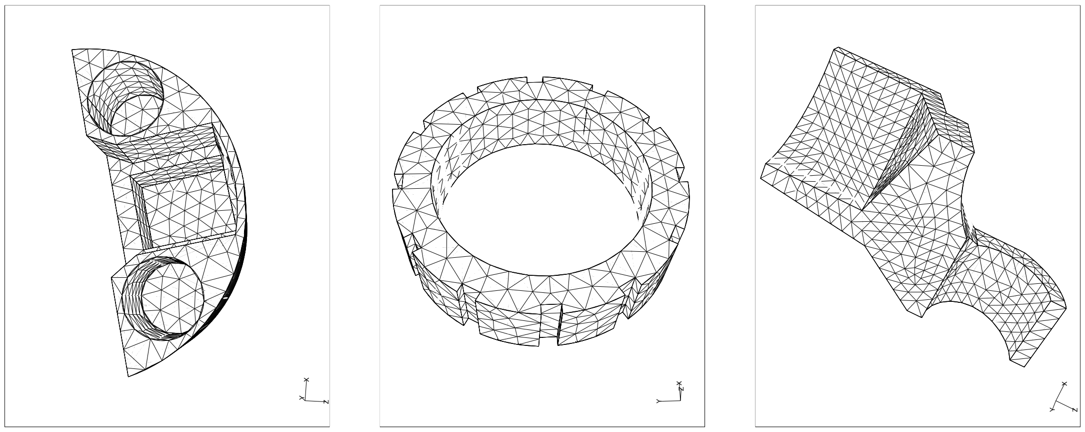

Figure 4: Surface meshes for the foam, gear, and hook meshes.

5 Numerical Results

AMPL [14] was used during the development of the inexact Newton code to verify the correctness of the analytic gradient and Hessian evaluations and to check that the solution computed by the inexact Newton code is consistent with the solution found by other general-purpose methods. Eleven meshes were used for testing purposes. The triangular meshes were obtained from the Triangle meshing package [35], and the tetrahedral meshes were obtained from CUBIT [33]. Figure 3 plots the deer and turtle triangular meshes; Figure 4 displays the surface mesh for the foam, gear, and hook tetrahedral meshes. Table 1 reports statistics for all of test meshes, including the number of vertices and elements in the mesh and the total number of variables once the boundary vertices have been removed. A feasible starting point is provided for all these example problems.

All tests in this section were run on a 1.8 GHz Intel Pentium 4 machine running Linux. The inexact Newton code was compiled by using gcc version 3.2 with the -O9 optimization flag. This machine has 512 MB of RAM with a 256 KB cache and was not running any other jobs at the time the results were generated. The codes were executed from a local

Table 1: Statistics for the test meshes

Mesh	Type	Vertices	Elements	Variables
deer	triangular	1,122	1,896	1,520
turtle	triangular	2,222	4,025	3,578
rand1000	triangular	1,152	2,170	2,048
rand10000	triangular	10,400	20,394	20,000
foam	tetrahedral	1,337	4,847	867
gear	tetrahedral	866	3,116	780
hook	tetrahedral	1,190	4,675	1,200
duct20	tetrahedral	1,067	4,104	1,146
duct15	tetrahedral	2,139	9,000	2,868
duct12	tetrahedral	4,199	19,222	6,906
ductbig	tetrahedral	177,887	965,759	425,952

disk and did not access any network disks.

The inexact Newton method computed solutions with $\|\nabla F(x)\|_2 \leq 1.0 \times 10^{-6}$ for all of the test meshes. Directions of negative curvature were obtained by the conjugate gradient method during several iterations of the inexact Newton algorithm on the rand1000 and rand10000 problems. Even though the objective function for the optimization problem is not convex, the Hessian matrix is positive definite at the critical point returned by the code for all the example meshes.

The choice of preconditioner used in the inexact Newton method is crucial when solving these optimization problems. When no preconditioner is used, the iteration limit in the conjugate gradient method is typically reached without computing a direction where the relative residual has decreased enough. The overall algorithm then fails to converge in 100 iterations. A diagonal preconditioner performs much better than no preconditioner. However, the diagonal preconditioner generally requires 5–15% more matrix-vector products than does the block Jacobi preconditioner.

The reordering can significantly reduce the time taken to solve the entire problem; the amount of the reduction is machine and problem dependent. Figure 5 presents the sparsity pattern for the Hessian matrix of the original and reordered meshes for the ductbig problem. For this mesh, the reduction in total solution time realized by reordering the vertices and elements was over 43%. The reason for this improvement is that most of the computational time for the inexact Newton code is in the Hessian evaluation and matrix-vector products, where locality of reference has a large impact. On the rand10000 mesh, the reduction in total time was over 40%. The inexact Newton method on the reordered mesh performs fewer function evaluations and conjugate gradient iterations than on the original mesh, accounting for some of this large decrease in total time. The reduction in time due to reordering on the other meshes is hard to quantify because the total time for solving the original and reordered meshes differs by only a few hundredths of a second. The cost of computing the reordering, however, may dominate the savings in the linear algebra, leading to an increase in total time. Such increases are realized when solving the test problems with fewer than 1,500 variables.

Having developed an AMPL model to solve the mesh shape-quality optimization problem

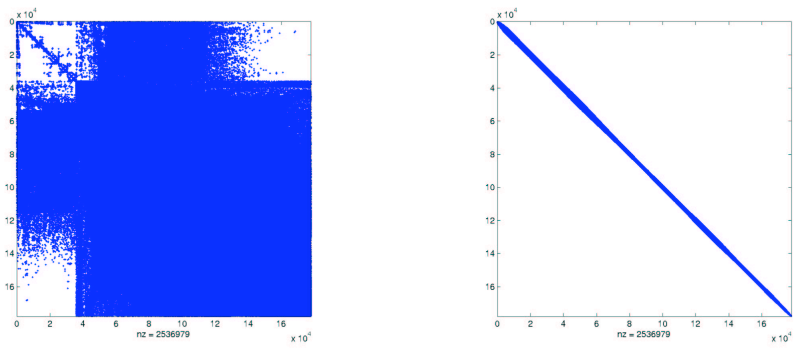

Figure 5: Sparsity pattern of the Hessian matrix for the ductbig mesh with original ordering (left) and the breadth-first search ordering (right).

and written a special-purpose code to solve the same problem, we now compare the two in order to quantify the effects on solution times and memory requirements. Two AMPL models were written for these tests: one is a constrained version of the optimization problem, and the other is an unconstrained formulation. The denominator in both AMPL models is computed by multiplying $\det(A)$ by the constant term $2\det(W^{-1})$. This modification does not affect the optimization problem, but reduces the amount of memory consumed by AMPL and the general-purpose solvers when optimizing the problem. Furthermore, two versions of each mesh were used for these numerical results; the original mesh is in the order provided by the meshing package, and a symmetric permutation has been applied in the reordered mesh, where the vertex and element ordering was calculated by the inexact Newton code. Both AMPL models and all the test meshes are available for download from `http://www.mcs.anl.gov/~tmunson/models`. Versions of these models are also available in the COPS 3.0 test set [13].

KNITRO 3.0 [41] and LOQO 6.06 [39] were used to solve the constrained formulation; and LOQO, KNITRO, and TRON [25] were used to solve the unconstrained problems. All of these solvers use exact Hessian matrices and were run with their default settings. Each solve was attempted three times, with the minimum time taken over these runs reported. The time command was used to obtain the clock time from start to finish for the particular codes, and the user time is reported.

The time for the general-purpose algorithms includes the time taken by AMPL to read the model and data files, generate the problem, and read the solution file produced by the solver. The optimized mesh was *not* written to disk for the AMPL models, and the time to reorder the mesh is not included, since this calculation was performed exogenously. The time for the inexact Newton method includes reading the mesh, computing the sparsity pattern for the Hessian, solving the problem, and writing the optimized mesh to disk. The cost of computing the reordering for the mesh is included in the total time for the inexact Newton method when a reordering was used.

Table 2 reports the time taken in seconds to solve the original and reordered problems using the constrained formulation, while Table 3 reports the time taken in seconds on the unconstrained formulation. These tables also report the amount of RAM used by the codes

Table 2: Results for meshes when using the constrained AMPL model.

Mesh	Ordering	LOQO	KNITRO	Inexact Newton
deer	Original	†	1.25	0.05
	Reordered	†	1.12	0.04
turtle	Original	33.59	4.33	0.14
	Reordered	†	4.00	0.12
rand1000	Original	12.79	‡	0.18
	Reordered	2.76	‡	0.16
rand10000	Original	80.95	‡	7.44
	Reordered	57.05	‡	4.43
foam5	Original	6.53	4.70	0.10
	Reordered	6.11	4.50	0.12
gear	Original	3.75	4.28	0.07
	Reordered	3.45	4.00	0.07
hook	Original	8.86	7.25	0.10
	Reordered	8.79	6.99	0.11
duct20	Original	11.63	6.12	0.09
	Reordered	10.42	5.64	0.07
duct15	Original	41.65	16.59	0.23
	Reordered	28.01	15.22	0.21
duct12	Original	112.00 (435 MB)	61.20 (443 MB)	0.58 (3.4 MB)
	Reordered	79.80 (426 MB)	51.94 (434 MB)	0.48 (3.5 MB)
ductbig	Original	↕	↕	86.28 (175 MB)
	Reordered	↕	↕	48.61 (180 MB)

† Iteration limit reached. ‡ Current solution could not be improved. ↕ Ran out of memory.

Table 3: Results for meshes when using the unconstrained AMPL model.

Mesh	Ordering	LOQO	KNITRO	TRON	Inexact Newton
deer	Original	1.31	0.59	0.76	0.05
	Reordered	1.21	0.55	0.60	0.04
turtle	Original	2.43	1.76	2.11	0.14
	Reordered	2.20	1.61	1.73	0.12
rand1000	Original	2.95	†	†	0.18
	Reordered	2.44	†	†	0.16
rand10000	Original	49.21	†	†	7.44
	Reordered	42.78	†	†	4.43
foam5	Original	4.51	2.87	2.79	0.10
	Reordered	4.45	‡	2.72	0.12
gear	Original	2.59	‡	2.60	0.07
	Reordered	2.48	‡	2.31	0.07
hook	Original	8.71	3.30	3.82	0.10
	Reordered	7.27	3.23	3.73	0.11
duct20	Original	16.15	3.50	4.02	0.09
	Reordered	16.00	3.20	3.37	0.07
duct15	Original	40.42	9.27	10.74	0.23
	Reordered	38.75	8.39	9.78	0.21
duct12	Original	127.93 (375 MB)	†	29.20 (369 MB)	0.58 (3.4 MB)
	Reordered	134.06 (374 MB)	†	24.20 (367 MB)	0.48 (3.5 MB)
ductbig	Original	↕	↕	↕	86.28 (175 MB)
	Reordered	↕	↕	↕	48.61 (180 MB)

† Function undefined. ‡ Current solution could not be improved. ↕ Ran out of memory.

for the duct12 and ductbig problems. Note that KNITRO and TRON compute a trial point where the objective function is undefined for some of the unconstrained problems. Their AMPL links report an error instead of returning a domain violation or plus infinity for the objective function value.

These results demonstrate, as expected, that using a special-purpose code can be significantly faster than using a general-purpose algorithm within a modeling environment and can require fewer memory resources. For example, on the duct12 problem, the TRON algorithm on the unconstrained formulation with the reordered mesh computed an optimal solution in the shortest amount of time. However, this solve took over 50 times longer to compute a solution than did the special-purpose code and consumed more than 100 times the memory. The other solvers were over 100 times slower or encountered a failure. In general, using a general-purpose solver from within AMPL is approximately 9.5 to 275 times slower than using the special-purpose code over all of the test meshes and formulations for the shape-quality optimization problem. Moreover, the general-purpose codes can fail to remain feasible on the constrained optimization problems even though they are started from a feasible point, particularly on the triangular meshes.

A more surprising outcome is the effect that applying a symmetric permutation to the optimization problem can have on the general-purpose solvers. Algorithms, such as LOQO, that apply direct methods to compute the directions may not see any benefit from applying the permutation because the factorization routine should reorder the matrices for sparsity. However, the permutation effects may be more pronounced for solvers, such as TRON, that rely on iterative techniques to compute the directions, because of better locality of reference in the matrix-vector products.

In general, the breadth-first search ordering reduces the time to compute a solution by all of the general-purpose codes tested. One of the reasons for this outcome is that the permutation can reduce the amount of time taken by the AMPL solver library to compute the function, gradient, and Hessian evaluations. Sometimes the result is dramatic because of large fluctuations in the total number of iterations taken to solve the two problems. The most extreme case observed is when the solver fails to compute a solution for the reordered problem but succeeds for the original ordering of the same problem, such as LOQO on the turtle mesh and KNITRO on the foam5 mesh.

TRON derives a consistent improvement in solution time from the reordering; the least improvement is a 2.35% reduction in time on the hook mesh, while the maximum is a 21% reduction on the deer mesh. KNITRO also obtains some improvements in solution time, although the reduction in time is generally under 10%. In particular, the least improvement in time for KNITRO is a reduction of 2% on the unconstrained formulation of the hook mesh, while the best is a 15% reduction on the constrained formulation of the duct12 mesh.

The effect of the reordering on LOQO is hard to quantify. LOQO experiences a 78% improvement in solution time on the constrained version of the rand1000 mesh, primarily because of a significant reduction in the number of iterations taken to solve the problem. The same situation arises on the rand10000 mesh. On several meshes, the reduction in time is under 1%. Furthermore, the reordered problem takes longer to solve than the original problem for the unconstrained version of the duct12 mesh when using LOQO.

6 Conclusion

The mesh shape-quality optimization problem is an application where performance is critical, especially when r-refinement is used in conjunction with an adaptive scheme or when the mesh changes through time. The savings in memory and computational time by using a special-purpose algorithm versus using a general-purpose algorithm within a modeling language can be large. In particular, the general-purpose algorithms are between 9.5 and 100 times slower when used through the AMPL modeling language than the special-purpose code on our optimization problems and can consume over 100 times the memory.

To improve the performance of the special-purpose code, we used a block Jacobi preconditioner within the conjugate gradient method. We proved that this preconditioner is positive definite by applying a theorem on the convexity of fractional functions to the inverse mean-ratio metric and showing that the appropriate components of the Hessian matrix are invertible. The theorem on the convexity of fraction functions may be useful in other situations and for different quality metrics.

The results obtained from applying a symmetric permutation to the optimization problem were surprising; this modification can increase or decrease the number of iterations taken to solve the problem, and the algorithms can even fail to solve the permuted model. However, the reordering generally reduces the total time required to compute a solution. These observations may be useful in improving the efficiency and robustness of general-purpose algorithms.

Acknowledgments

Lori Freitag-Diachin, Patrick Knupp, and Suzanne Shontz introduced me to the mesh shape-quality optimization problem and the inverse mean-ratio metric when they were working on a summer project and asked whether I knew of an efficient algorithm to solve their problems. This work is a direct consequence of my collaborations with them.

David Gay provided both the elegant version of the area calculation used in the AMPL model for three-dimensional meshes and the idea for the unconstrained models. Paul Hovland wrote the initial version of the analytic gradient of the element function that uses a minimal number of floating-point operations. The turtle outline was created by Elizabeth and Jacob Dolan and the deer outline by Joshua Griffin; these outlines were translated into a geometry and meshed by Suzanne Shontz.

Mihai Anitescu helped with an early version of the proof of Lemma 3.7 by providing the idea for some of the reductions applied to the optimization problem.

References

[1] L. Armijo. Minimization of functions having Lipschitz-continuous first partial derivatives. *Pacific Journal of Mathematics*, 16:1–3, 1966.

[2] I. Babuška and M. Suri. The p and h-p versions of the finite element method, basic principles and properties. *SIAM Review*, 36:578–632, 1994.

[3] T. J. Baker. Mesh movement and metamorphosis. In *Proceedings of the Tenth International Meshing Roundtable*, pages 387–296. Sandia National Laboratories, 2001.

[4] R. E. Bank and R. K Smith. Mesh smoothing using a posteriori estimates. *SIAM Journal on Numerical Analysis*, 34(3):979–997, 1997.

[5] M. Berzins. Solution-based mesh quality for triangular and tetrahedral meshes. In *Proceedings of the Sixth International Meshing Roundtable*, pages 427–436. Sandia National Laboratories, 1997.

[6] M. Berzins. Mesh quality – geometry, error estimates or both? In *Proceedings of the Seventh International Meshing Roundtable*, pages 229–237. Sandia National Laboratories, 1998.

[7] C. H. Bischof, P. D. Hovland, and B. Norris. Implementation of automatic differentiation tools. *Higher-Order and Symbolic Computation*, to appear, 2004.

[8] S. C. Brenner and L. R. Scott. *The Mathematical Theory of Finite Element Methods.* Springer-Verlag, New York, 2002.

[9] A. Brooke, D. Kendrick, and A. Meeraus. *GAMS: A User's Guide.* The Scientific Press, South San Francisco, California, 1988.

[10] R. Byrd, M. E. Hribar, and J. Nocedal. An interior point method for large scale nonlinear programming. *SIAM Journal on Optimization*, 9:877–900, 1999.

[11] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 24th National Conference ACM*, pages 157–172. ACM Press, 1969.

[12] C. Ding and K. Kennedy. Improving cache performance in dynamic applications through data and computation reorganization at run time. In *Proceedings of the 1999 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pages 229–241, 1999.

[13] E. D. Dolan, J. J. Moré, and T. S. Munson. Benchmarking optimization software with COPS 3.0. Technical Memorandum ANL/MCS-TM-273, Argonne National Laboratory, Argonne, Illinois, 2004.

[14] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming.* Brooks/Cole–Thomson Learning, Pacific Grove, California, second edition, 2003.

[15] L. Freitag and P. Knupp. Tetrahedral mesh improvement via optimization of the element condition number. *International Journal for Numerical Methods in Engineering*, 53:1377–1391, 2002.

[16] L. Freitag, P. Knupp, T. Munson, and S. Shontz. A comparison of optimization software for mesh shape-quality improvement problems. In *Proceedings of the Eleventh International Meshing Roundtable*, pages 29–40. Sandia National Laboratories, 2002.

[17] L. Freitag and C. Ollivier-Gooch. A comparison of tetrahedral mesh improvement techniques. In *Proceedings of the Fifth International Meshing Roundtable*, pages 87–100. Sandia National Laboratories, 1996.

[18] L. Freitag and C. Ollivier-Gooch. A cost/benefit analysis for simplicial mesh improvement techniques as measured by solution efficiency. *International Journal of Computational Geometry and Applications*, 10:361–382, 2000.

[19] L. Freitag and P. Plassmann. Local optimization-based simplicial mesh untangling and improvement. *International Journal for Numerical Methods in Engineering*, 49:109–125, 2000.

[20] A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation.* SIAM, Philadelphia, Pennsylvania, 2000.

[21] H. Han and C. Tseng. A comparison of locality transformations for irregular codes. In *Proceedings of the Fifth International Workshop on Languages, Compilers, and Run-time Systems for Scalable Computers*, pages 70–84, Rochester, New York, 2000. Springer-Verlag.

[22] C. T. Kelley. *Solving Nonlinear Equations with Newton's Method.* SIAM, Philadelphia, Pennsylvania, 2003.

[23] P. Knupp. Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities, Part I – A framework for surface mesh optimization. *International Journal for Numerical Methods in Engineering*, 48:401–420, 2000.

[24] P. Knupp. Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities, Part II – A framework for volume mesh optimization and the condition number of the Jacobian matrix. *International Journal for Numerical Methods in Engineering*, 48:1165–1185, 2000.

[25] C.-J. Lin and J. J. Moré. Newton's method for large bound-constrained optimization problems. *SIAM Journal on Optimization*, 9:1100–1127, 1999.

[26] A. Liu and B. Joe. Relationship between tetrahedron quality measures. *BIT*, 34:268–287, 1994.

[27] O. L. Mangasarian. *Nonlinear Programming.* McGraw-Hill, New York, 1969. SIAM Classics in Applied Mathematics 10, SIAM, Philadelphia, 1994.

[28] T. S. Munson. Mesh shape-quality optimization using the inverse mean-ratio metric: Tetrahedral proofs. Technical Memorandum ANL/MCS-TM-275, Argonne National Laboratory, Argonne, Illinois, 2004.

[29] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, New York, 1999.

[30] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press, San Diego, California, 1970.

[31] R. T. Rockafellar. *Convex Analysis.* Princeton University Press, Princeton, New Jersey, 1970.

[32] Y. Saad. *Iterative Methods for Sparse Linear Systems.* SIAM, Philadelphia, Pennsylvania, second edition, 2003.

[33] Sandia National Laboratories, Albuquerque, New Mexico. *CUBIT 8.1 Mesh Generation Toolkit*, 2003.

[34] M. Shephard and M. Georges. Automatic three-dimensional mesh generation by the finite octree technique. *International Journal for Numerical Methods in Engineering*, 32:709–749, 1991.

[35] J. Shewchuk. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Proceedings of the First Workshop on Applied Computational Geometry*, pages 124–133, Philadelphia, Pennsylvania, May 1996. ACM.

[36] J. Shewchuk. What is a good linear element? Interpolation, conditioning, and quality measures. In *Proceedings of the Eleventh International Meshing Roundtable*, pages 115–126. Sandia National Laboratories, 2002.

[37] S. M. Shontz and S. A. Vavasis. A mesh warping algorithm based on weighted Laplacian smoothing. In *Proceedings of the Twelfth International Meshing Roundtable*, pages 147–158. Sandia National Laboratories, 2003.

[38] L. N. Trefethan. *Spectral Element Methods in MATLAB.* SIAM, Philadelphia, Pennsylvania, 2000.

[39] R. J. Vanderbei. LOQO user's manual – Version 4.05. Technical report, Princeton University, Princeton, New Jersey, 2000.

[40] R. J. Vanderbei and D. F. Shanno. An interior-point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13:231–252, 1999.

[41] R. Waltz and J. Nocedal. KNITRO user's manual – Version 3.1. Technical Report 5, Northwestern University, Evanston, Illinois, 2003.